Presented 19 June 2001 at the
6th International Command & Control Research & Technology Symposium
USNA, Annapolis, MD; 19 – 21 June 2001

An Enterprise Identifier Strategy for Global Naming Across Arbitrary C4I Systems

Sam Chamberlain

U.S. Army Research Laboratory
ATTN: AMSRL-CI-CT
Aberdeen Proving Ground, MD 21005-5066
410-278-8948
wildman@arl.army.mil
http://www.arl.army.mil/~wildman

Abstract

In any information system, a critical feature is the ability to link together disparate pieces of data and information via relationships. One way to greatly facilitate this task is to provide a common identification technique that allows data and information items be conveniently and uniformly referenced. This can be accomplished by standardizing one field across disparate data sources. Perhaps no simpler enhancement can produce such a huge benefit with as little intrusion into legacy systems. This is the objective of enterprise identifiers. If data can be globally identified using a common scheme, then one can spontaneously reference, and ultimately, "plug and play" disparate, arbitrary pieces of information without prior coordination.

1. Introduction

In 1997, the question was asked: "With all the new information technology available, why is interoperability so difficult to achieve?"[1] This discussion lead to a CCRP (C4ISR Cooperative Research Program[2]) study [2] that produced two main conclusions: first, no central theme has been identified to facilitate the integration of disparate information components, and second, no common naming convention has been established. As a result, two concepts were developed and proposed to assist in filling these voids. First, as a central theme for information integration, the notion of default operational organizations was proposed that identifies force structure as the underlying theme by which all battle command operations, functions, and tasks are ultimately associated. Second, a simple naming convention for force structure components was developed, called organization identifiers (Org-ID), that is void of any information about the entities being identified.

With the aid of several other contributors, these concepts have grown and evolved naturally into more general processes that address a wider range of issues; see [1]. An objective was to provide simple and basic techniques that offer a large pay off for integration, yet are as unintrusive as

[1] Discussion with Dr. Dave Alberts at NDU, 14 Nov 1997.
[2] CCRP, see http://www.dodccrp.org/;
C4ISR: Command, Control, Communications, Computers, Intelligence, Surveillance, and Reconnaissance

possible so that they can be applied to legacy, as well as new, systems. The topic of force structure as a central theme for integration has evolved into the subject of *stationary data*. Stationary data is information that is "semi-static," meaning that it is relatively invariant over its lifetime and its periodicity is known and of significant duration to allow it to be reasonably maintained in a shared reference library. An example of stationary data is phone numbers that are maintained in a reference library called a phone book. The topic of Org-IDs has evolved into the subject of *enterprise identifiers* for all data, not just force structure. An enterprise identifier is a discriminator that uniquely distinguishes an information item across the entire enterprise, not just within a specific domain. If the enterprise is the U.S. Department of Defense (DoD), then its enterprise identifiers uniquely identify data across all participating databases in the DoD.

The topics of stationary data and enterprise identifiers are described in a recent report entitled *Enterprise Identifiers For Logistics - An Approach in Support of Army Transformation Initiatives* [9]. However, the implementation approach proposed in this study (and in [2]) has since been determined to be overly constraining and too dependent on the success and schedules of other programs. Consequently, a revised approach is being developed that is simpler, more streamlined, and most importantly, more direct. This paper describes the basic notions behind stationary data and enterprise identifiers and presents a revised approach for allocating and managing both enterprise identifiers and stationary data.

2. Enterprise Identifiers (EID)

An identifier is a property that discerns something. Within computers, different techniques may be used to accomplish the task of identification, but regardless of the technology, there must be some way to identify and reference the stored data. Databases are commonly used to store data and they may be based on one of several models. The two most common models are the relational and object-oriented models that identify items using *primary keys* (PK) and *object identifiers* (OID), respectively. A major accomplishment towards interoperability would be the acceptance of a common identification scheme that spans both of these data storage technologies.

In the relational model, data is stored in tables with attributes (i.e., columns) and the rows of a table may represent an entity or a relationship between entities. Every row of a table must be uniquely identifiable. A *candidate key* is a set of attributes that accomplishes this task. There may be several candidate keys for a table, and one of these is selected as the PK for the table. A consequence of this approach is that a PK may be composed of several attributes. Further, these attributes may be imported from other tables (called foreign keys, or FK), they may contain codes or symbols that provide insight into the properties of the item they identify, and the structure of the PK may be different for every table. Although some designers may view these characteristics as "features", they introduce significant inertia and complexity when one attempts to modify or share PKs across disparate databases. This is referred to as the "foreign key ripple effect" in [5].

To alleviate some of these problems, the concept of surrogate keys was introduced [3],[4],[8]. A *surrogate key* (SK) is a PK that is composed of a single attribute with no intelligence encoded into it. These constraints mean that the SK can not be composed of parts imported from another table and that the value of the SK provides no insight about the item it identifies. However, a SK only

needs to be unique within a single table. The same SK may exist in numerous tables (e.g., a common approach is to number rows from 1 to N in every table[3]).

Enterprise keys expand upon this concept. An enterprise key (EK) is a surrogate key that is unique across all tables of the enterprise [6]. Therefore, if the enterprise were the DoD, then every row of every table of every database in the DoD would be uniquely identified via an EK. This is a powerful property when the challenging task of integration is pursued.

An *enterprise identifier* (EID) is the generic version of an EK. A goal is to develop an identification strategy that is independent of any database technology or schema. An EID is special because it uniquely identifies an entity or object across the entire enterprise, not just within a specific domain or type of database. Similarly, if two data items have the same EID, then they must be semantically equivalent; that is, they must represent the same thing. This allows different formalisms to be used to represent identical items. It does not matter if the item is represented as a row in a relational database (with an EK) or as an object in an object database (with an Object ID); if two items have the same EID, then they are semantically equivalent.

The two primary challenges to accomplish an EID implementation are (1) the development of a flexible, high-performance EID allocation scheme that can be as distributed or centralized as desired, and (2) reaching agreement on a common structure usable by any database technology.

2.1 EID Allocation Scheme

A primary challenge in creating an EID allocation system is developing an approach to guarantee that EIDs are unique while ensuring that they can be easily obtained. In the Internet community, this is being accomplished with Uniform Resource Identifiers (URIs).[4] However, for databases, and particularly databases communicating using low bandwidth environments, the URI approaches are verbose and of widely variable in size. Consequently, a type of URI for database applications is required. Whenever a database management system creates a piece of data (e.g., inserts a row into a table), it must obtain a globally unique EID to tag the data. This must be accomplished rapidly because any significant delay is unacceptable. Further, all EIDs must have the same structure and be sized to be large enough to handle the vast number of required cases without being excessively large to cause performance degradation or a communications burden on C4ISR systems.

The recommended size for an EID is 64 bits. This provides 18 billion billion combinations[5], or in other words, it allows 4.3 billion systems to each create 4.3 billion data items. Because EIDs have no information encoded in them, they do not have to be "blocked" into subparts as is required by Internet addresses or phone numbers. Therefore, 64 bits provides a name space large enough to handle a sizable data tagging task while still being compact enough to be used over low

---

[3] As occurs with the automatic indexing feature found in many databases.

[4] For URI Working Group Charter, see http://www.ietf.cnri.reston.va.us/proceedings/94mar/charters/uri-charter.html.

[5] Actual value is 2^{64} combinations which is 18.44×10^{18}, or 18.44 quintillion.

bandwidth, tactical communication systems. However, the top bit of the EID can always be reserved to allow future expansion should it become necessary.

A common approach for producing globally unique values is to concatenate a locally managed unique value to an existing globally unique value. In other words, a large globally unique value can be created by combining two smaller unique numbers. If one of the smaller numbers is a globally unique value assigned to only one subscriber, then that subscriber can append a locally managed unique number to it and still produce a globally unique value. Thus, an EID can be constructed to globally identify arbitrary data by appending a locally managed number to a pre-allocated globally unique value called an *EID Seed*. This is illustrated in Figure 1.

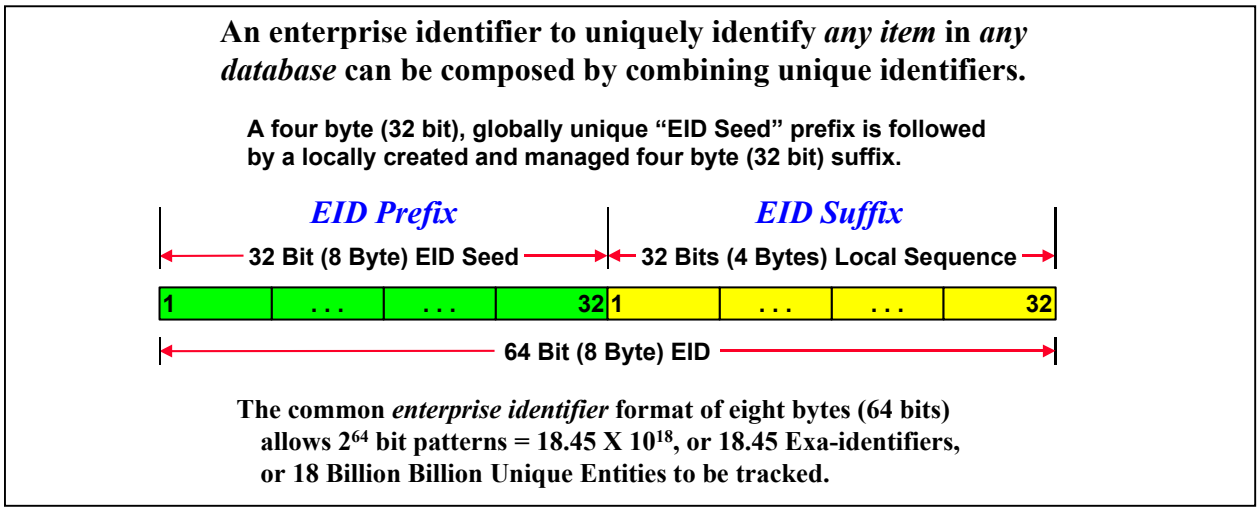

Figure 1: EID Composition

2.2 EID Allocation Architecture

Figure 2 illustrates the EID allocation architecture. The architecture includes three levels of components: EID users (on the right), EID servers (in the middle), and EID seed servers (on the left). Any time a user creates data, it must be uniquely tagged with an EID. This is accomplished by obtaining an EID from any ES. An *EID server* (ES) is any computer program that provides EIDs to requestors. As illustrated in Figure 1, an ES creates a 64 bit EID by appending a locally generated 32-bit EID suffix to a 32-bit EID prefix that is an EID seed obtained from an *EID seed server* (ESS). An ESS is a member of a special set of redundant servers that simply passes out EID seeds to registered users. Since an EID seed is a 32-bit value, 4.3 billion ESs may be established.[6] Because an EID suffix is also a 32-bit value, each ES may generate 4.3 billion EIDs for each EID seed. In other words, this technique supports 4.3 billion ESs, each capable of providing 4.3 billion EIDs, for a total of 18 billion billion EIDs. In Figure 2, the EID values are represented using hexadecimal notation where each character (with a value of '0'–'9' or 'A'–'F') represents a sequence of four bits. Therefore, a 64-bit EID is denoted by 16 characters.[7]

---

[6] The actual value is 2^{32} ESs, which is 4.295 billion.

[7] Each of the 16 values ('0' – 'F') denotes one of the 16 combinations of four bits: 0000 through 1111. The fact that 16 characters are also required to denote a 64-bit EID is a coincidence.

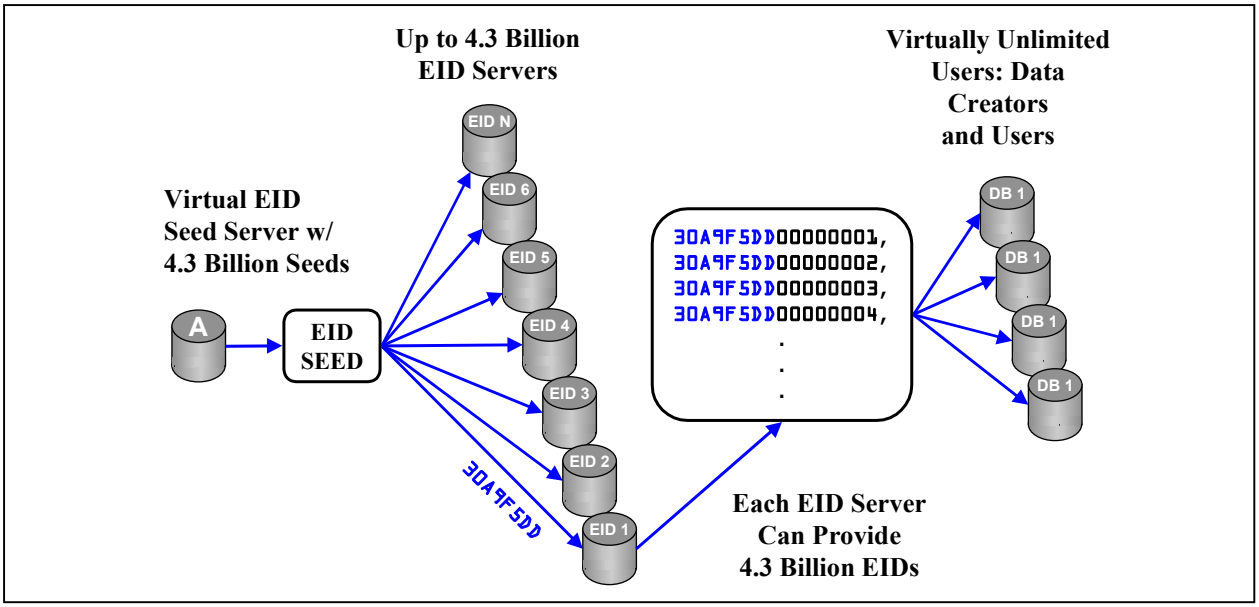

Figure 2: EID Allocation Architecture

2.2.1 EID Seed Servers

ESSs are a special set of controlled servers that pass out 32-bit EID seeds to registered users. They form a virtual server because, externally, they appears to be a single resource although several tightly coupled, redundant machines are use to provide robustness, rapid speed of service, and backup. To establish an ES, one must obtain an EID seed from an ESS and this requires an ESS account. Obtaining an account is not a difficult procedure, but it is required. Anyone may request an account, but ultimately, accounts may be scrutinized to those with a bona fide requirement or official capacity to create official government data. Example are officials in corporate information offices (CIO), acquisition organizations, such as program executive officers (PEO), project managers (PM), and systems builders, and the simulation community. To handle non-traditional cases, a human will be required in the authorization process. The requirement is to respond to a request within 24 hours, although the goal is within minutes.

Conceptually, registration would occur via a browser interface over a secure connection. To register, a person would have to provide a viable point of contact (POC) and the position of the POC (e.g., Assistant PM for the XYZ systems) or the reason for the account. There would be a default set of restrictions on the number or rate of EID seeds that can be obtained. For example, there may be restrictions on the maximum total number of EID seeds that can be obtained without scrutiny, the maximum allowed for one request, or a maximum allowed per day, etc. During the registration process, the requestor would be presented the default set of restrictions. To exceed these values, a justification would be required. Finally, a list of rules that must be followed to establish and maintain an ES would be presented to the requestor. When the requestor accepts these conditions, the request is passed on to a human for final approval. Within 24 hours, the account request is granted or denied and the requestor is sent an account login identifier and an initial password.

Once an ESS account is obtained, the account owner can obtain EID seeds and establish ESs for use by data creators. Recall that one EID seed can be used to create over 4.3 billion EIDs. Therefore, large blocks of EID seeds would only be required for cases of highly distributed and autonomous systems (e.g., warrior and on-board command & control and weapon systems).[8]

An ESS can provide other services in addition to allocating EID seeds. One potential service is a lookup service. Recall that a primary advantage of EIDs is that they facilitate the creation of arbitrary associations between disparate data located within independent databases. In this environment, it is inevitable that one will occasionally receive an EID that references data to some unknown system outside its own.[9] For these cases, the ESS can provide a lookup service provided that it tracks to what account an EID seed is accepted. To accomplish this feature, an account owner would be required to provide a host address of a machine that will provide a lookup service for each EID seed received. The lookup service host does not have to be the ES that uses the EID seed. This is to allow for proxies in situations where one may not want to advertise its ES(s) due to security or other reasons. If an unknown EID is received, a lookup request is sent to an ESS, that in turn, returns the identity of the lookup service host for the unknown EID based upon its EID prefix (i.e., an EID seed). A repeat request can then be sent to the appropriate lookup server. Whether the ES lookup server responds to the request is a local decision.[10] It is emphasized that an ESS retains no information about an EID seed other than, one, the POC of the account that accepted the EID seed, two, the status of the ES associated with the EID seed (active or dormant), and three, the host address of the optional lookup service for EIDs built from the EID seed.

Another useful service is a validation service. This would allow an ES to contact the ESS to check that it is using a correct EID seed. Since a human may obtain EID seeds, it is plausible that an EID seed is entered incorrectly into an ES when it is established (or is being re-initialized after a malfunction). When an EID seed is requested, if one submits the actual ES hostname as the EID lookup service address, then the ESS can provide a validation service to an ES using a simple comparison operation. If a proxy is used, then the ESS can only report that the EID seed of the requesting ES belongs to the proxy. Thus, the proxy may request an EID seed validation from the ESS, and the ES would have to obtain validation from the proxy.

2.2.2 EID Servers

Once an EID seed is obtained, an ES may be established and begin passing out its 4.3 billion EIDs. Upon accepting an ESS account, an ES establisher agrees to adhere to two strict constraints: first: all EIDs created by the ES must be a 64 bits sequence composed of the bona fide, 32-bit EID seed prefix allocated to it; and two, the ES must ensure that the EIDs it produces are never duplicated. In other words, it must ensure that its locally generated 32 bit suffix is

---

[8] For example, a PM may be deploying thousands of systems that each require an autonomous data creation capability. This can be accomplished in several ways, to include providing individual EID seeds to each system, or by dividing up the 4.3 billion EIDs producible from a single EID seed.

[9] This should be a rare event in a tactical system where rigorous control is maintained over the data.

[10] This is analogous to Internet domain name service requests. Not all IP addresses are registered. This is the prerogative of the IP address owner.

always unique (i.e., it never allocates the same suffix twice). Other than the two constraints just stated, the ES owner has significant autonomy in how the ES is implemented.

The ES owner decides most implementation details, such as who is allowed to access the ES, how access is protected, and how EIDs are allocated. Access to an ES can be categorized as *single-user* or *multi-user*. A single user ES limits access to a single program, machine, or user. This may be the preferred approach for isolated systems, such as those with individual warriors (e.g., forward observers or special operations teams). A multi-user ES allows a variety of users and may be *open*, meaning anyone may obtain EIDs from it, or *restricted*, meaning that one requires permission to obtain EIDs. The ES owner decides whether protections, such as passwords or link encryption, are required.

The ES owner specifies how EIDs are allocated. The scheme does not need to be sophisticated. For example, a simple technique is to merely add one to the previous EID suffix for every EID request (i.e., 1, 2, 3, … , 4.3 billion). Such a mechanism would require just a few lines of Java code to implement a single-user, embedded ES like that for a PDA.[11]). At the other extreme, a large, multi-user ES may use more sophisticated EID allocation schemes. For example, the EID suffix space may be divided into segments so that each user has it own block of EID values. However, regardless of the allocation scheme used, to meet the uniqueness criteria, some type of persistent backup is required. This is to ensure that duplicate EIDs are never produced as a result of an ES malfunction or system fault. The details of how this is accomplished are a local implementation decision.

Like the ESS, an ES may offer a tracking service. To do this, the ES must maintain a record of an EIDs destination. For a single-user ES this is a trivial task since it is providing EIDs to only one destination. For a multi-user ES, this requires additional functionality akin to the ESS.[12] In any case, the ES owner always retains control over its operation. It is perfectly permissible to limit responses to lookup requests based on any criteria deemed appropriate by the ES manager.

2.2.3 EID Server Location

The elegance of this architecture is its flexibility. When an automated information system (AIS) creates data, it must obtain an EID from an ES. The location of the ES in relation to the AIS will depend upon policy, performance, and security issues; they do not have to be collocated. An ES may be embedded within an AIS, they may be located onboard a common platform (e.g., computer), or may be distributed via a local or wide area network (LAN, or WAN, respectively). These cases are illustrated in Figure 3.

Once again, performance will probably be the driving force behind the selected configuration. Certainly, isolated computer systems, like those carried by soldiers or on combat vehicles, will have onboard ESs. Command posts and tactical operation centers could choose onboard or redundant ESs on a LAN. Highly controlled, tightly coupled, but low volume applications may

---

[11] PDA – Personal Digital Assistant, for example, a Palm Pilot.®

[12] At this point, one should recognize the striking similarity between an ESS and a multi-user ES. They are essentially equivalent with the only difference being they size of the EID they provide (i.e., 32 versus 64 bits).

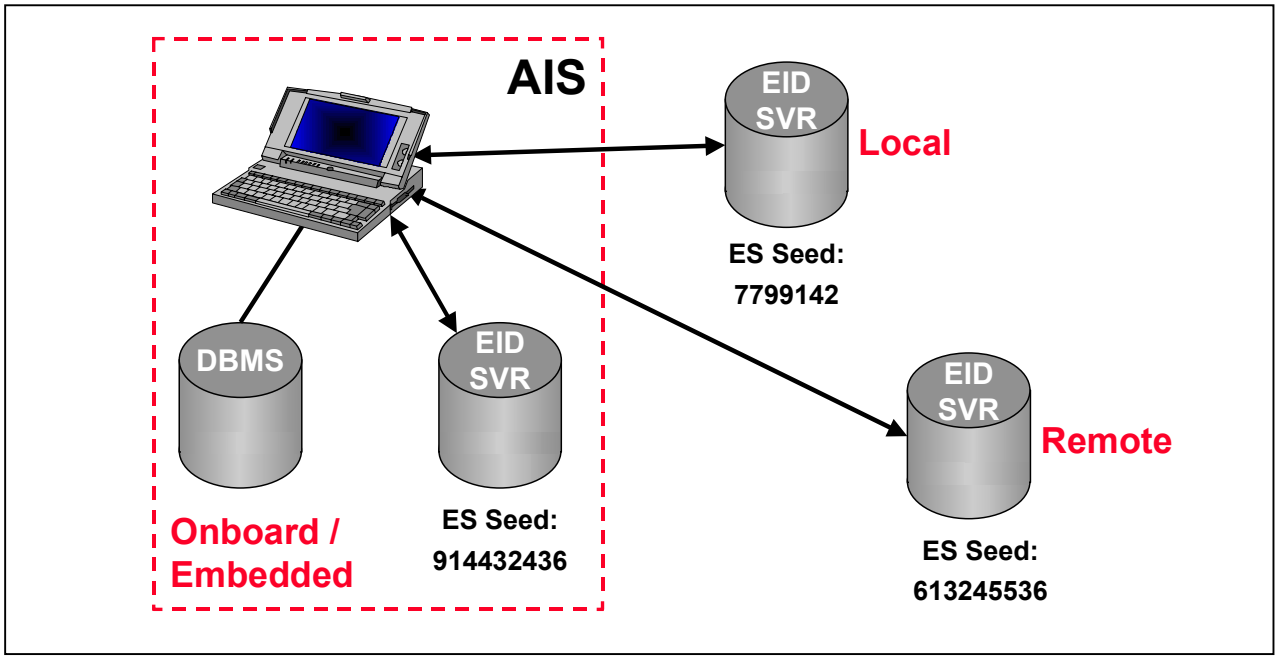

Figure 3: EID Server Location Options

choose to access a common ES via a secure WAN. Configuration control is immensely flexible and the options are boundless. The decision is the system managers.

Consider some extreme cases. Figure 4 illustrates the case where a centralized, multi-user ES is collocated with a centralized AIS. Theoretically, there could be one big AIS run by the DoD and every organization in the enterprise using the AIS would obtain EIDs from a collocated ES. At the other extreme, there could be 4.3 billion independent AISs, each with its own collocated ES. These might be associated with 4.3 billion people, each with their own wearable AIS. This is illustrated in Figure 5. Of course, an AIS does not have to be collocated with the ES it uses, and if a high-speed network connects them, performance would not suffer. Figure 6 illustrates a centralized AIS that uses many ESs. In this case, each user has their own ES. As they insert data into the centralized AIS they provide their own EIDs. Conversely, Figure 7 illustrates a set of distributed AISs that share a common ES. To insert data into their local AIS, they must obtain an EID from the central ES. The central ES may also include functions to check that other constraints are satisfied. For example, this might be a case where people are responsible for a portion of a distributed database and the central ES can check that they create only the data that they are authorized. Although these four examples show extreme cases, real configurations will be in the middle of these extremes. The advantage is that each system builder may implement their ES configuration based upon their own local policy, procedures, and performance requirements.

In summary, an EID server:

1. Must provide 64 bit EIDs (or its hexadecimal equivalent) using a bona fide EID Seed prefix.
2. Must never duplicate an EID. This implies that it must have some type of backup scheme to prevent re-use in case there is a loss of power or major malfunction.

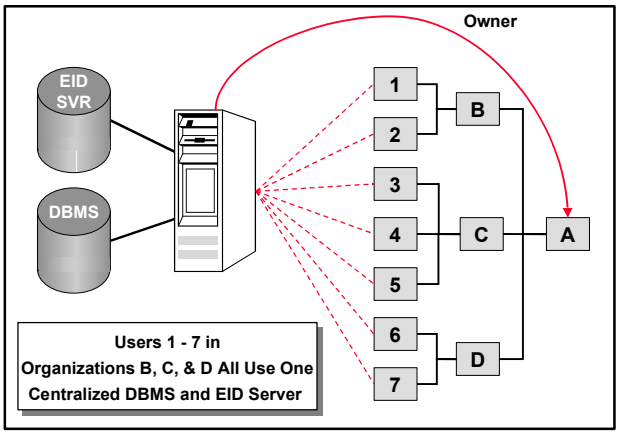

Figure 4: Extreme Case – Centralized

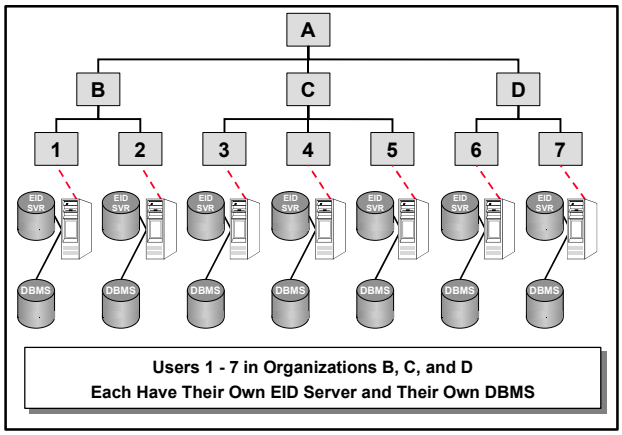

Figure 5: Extreme Case – Decentralized

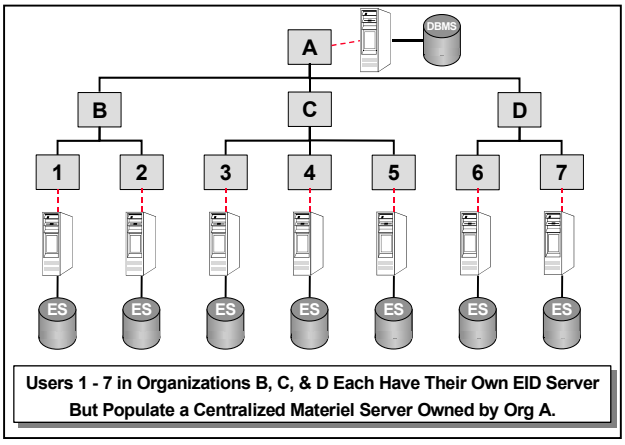

Figure 6: Mixed Extreme One

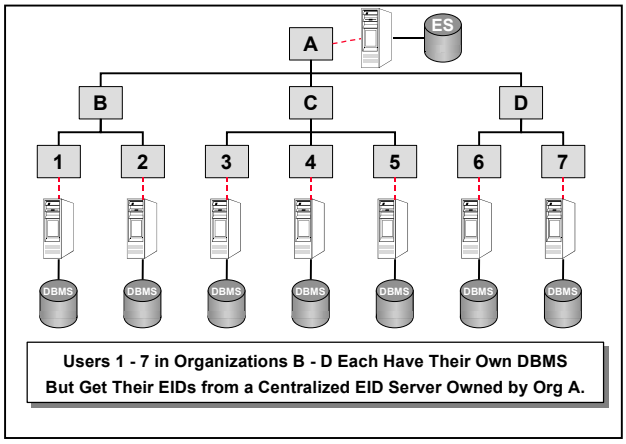

Figure 7: Mixed Extreme Two

3. Must be established by an authorized person who is registered with the EID Seed Server [Note that *the enterprise* is now defined as the set of users of the ESS and is not constrained by service, governmental, or national borders.]

Other policies concerning features, access and security are determined by the ES owner. Features such as optional services (such as a tracking service), encrypted links, login passwords, and single versus multi-user functionality are all local decisions. The goal is to maximize flexibility via local control and minimize bureaucracy once an EID seed is obtained..

3. Stationary Data

One way to categorize data is by its perishability. *Reference data* is static data. Examples are lookup table entries for state, airport, or country codes. At the other extreme is *dynamic data*. As its name implies, it is constantly changing and must be updated frequently to remain synchronized. Situational awareness (SA) data is in this category. The third category lies between these two extremes. As defined in Section 1, *stationary data* is information that is "semi-static," meaning that it is relatively invariant over its lifetime and its periodicity is known and of significant duration to allow it to be reasonably maintained in a shared reference library. An example of stationary data is phone numbers. Although thousands of phone numbers are added, deleted, and

changed daily worldwide, it is not a frequent event for a person's phone number to change. Usually, a person's phone number is static for the duration of their tour of duty. Therefore, phone numbers can be reasonably maintained in a reference library, called a phonebook, that is only distributed once per year. The small subset of new, changed, and deleted phone numbers is handled via other means (i.e., directory assistance).

Several C4ISR oriented data models (DM), related to the C2 Core DM, include battlefield entities in five basic domains: organization, materiel, personnel, facility, and features. These five domains contain large amounts of information that can be considered as stationary. This means that the data can reside in common reference libraries, provided via data servers, that can be periodically downloaded into one's computers; see Figure 8. By rigorously controlling the update process of the stationary data, users can be confident that they have a consistent set of reference material preloaded into their computers.

EIDs can be used to provide a common naming scheme across the reference libraries (i.e., for both the reference and stationary data) so that the common data will include a common set of EIDs. Once users have downloaded a common set of reference materiel, they may refer to it by passing the terse EIDs instead of the bulky data, thus significantly reducing the bandwidth required to manage C4ISR information.

An even more significant benefit is achieved when the agencies responsible for the data (i.e., the proponents) maintain the servers so that the data is not only consistent, but timely and accurate as well. To accomplish this, a proponent for the data must be identified to rigorously control the creation, maintenance, and deletion of the data. A reasonable practice is to begin with the agencies that already maintain the data as part of their charter. For example, every service has a force structure development community that handles a wide variety of force structure documents. These agencies should be the owners of the stationary and reference data for organizations. The

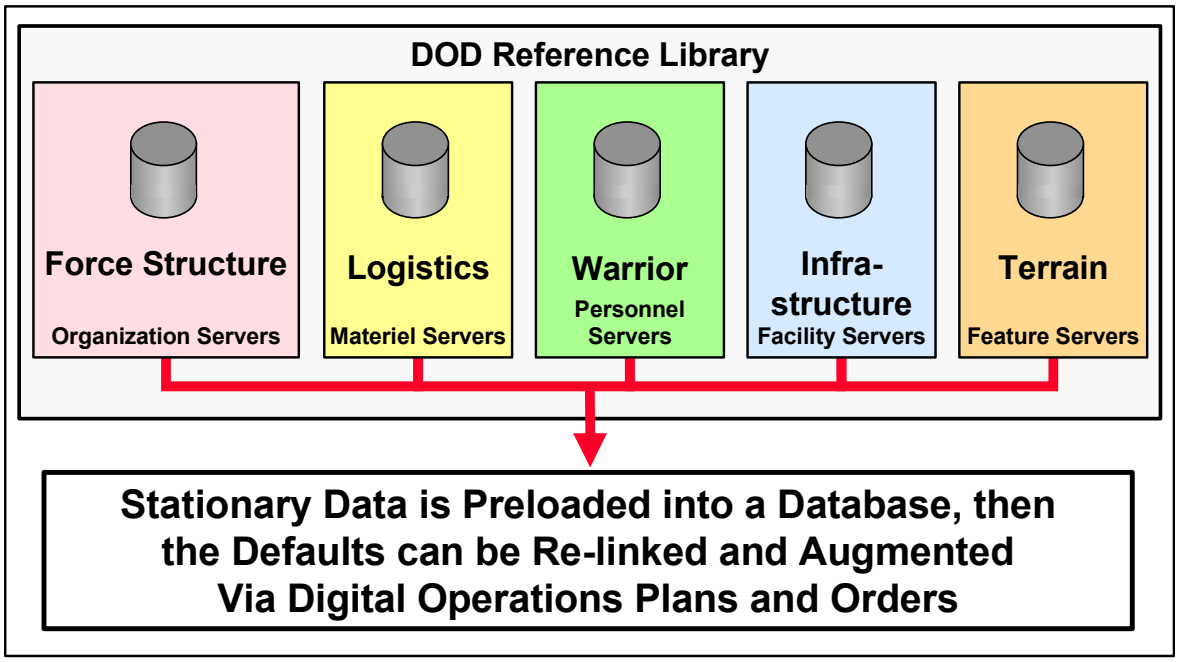

Figure 8: Suite of Servers that Contain Stationary Data

Defense Logistics Agency (DLA) already maintains the FEDLOG catalog via its Defense Logistics Information Service (DLIS)[13]. This could be the authoritative source for materiel related data that has national stock numbers (NSN). Note that there is a substantial amount of materiel that does not have NSNs. However, this does not preclude them from having EIDs. For example, the acquisition community (such as PMs) may create new materiel data (and assign EIDs) as a system evolves through the acquisition cycle. Several years may pass before a piece of equipment is type classified and receives an NSN, yet it may have had an EID for years. In other words, the authoritative source may not always be the data creator (and EID initiator).

There are many advantages to using meaningless values to identify data. Recall that two items with the same EID must be semantically equivalent. Consider static reference data, for example, country codes. Suppose that an official table exists that contains the official country codes for the world, one record for each country. Further, suppose that each country gets to pick what it wants to use for a two-character country code (a business key). Hidden from the user is the fact that each record is tagged with an enterprise-wide, unique EID. When an association with a record is required, or when machines exchange data, it is the EID that is of primary interest, not the two-character code. Although the United States may pick a two-character code of "US" for its official entry, other values can be attached to the same EID provided they "mean" the same thing. For example, different languages may be required, so an alias for "US" might be "EU" (in a Spanish system), or "VS" (in German), or "SU" (in Italian). As long as it is attached to the same EID that resides in the official table (the authoritative source), it will reference the same semantic entry (i.e., the country with the name "United States of America."). Because the EID has no information encoded into it, there is never a reason to change it. This general technique can be applied to any reference data (e.g., airport code) or stationary data (the name of a military organization). However, to provide a consistent product in an environment where the lines of ownership for reference and stationary data are rarely perfectly clear, it is important that authoritative sources be identified, sanctioned, and funded. For an excellent series of articles on the advantages of EIDs in relational database systems (i.e., the value of using enterprise keys), see [7].

4. EIDS – Past and Future

4.1 Original Allocation Scheme

The EID allocation scheme just described is slightly different from the originally scheme proposed in [2] and [9]. Previously, the EID seed was an Org-ID. To obtain an Org-ID, one had to be entered into an organization server by the force development community. Once that was accomplished, an organization could establish an EID server using its 32-bit Org-ID as the EID prefix. From that point on, the two schemes are identical. This allowed every organization, down to the billet level, to create 4.3 billion data items.

However, this approach had several negative ramifications. First, it tightly couples the ability to create data with the task of obtaining official recognition as an organization by the force development community. Although this may be considered a good feature in some communities,

[13] See http://www.dlis.dla.mil/govord.htm for information.

it adds another level of bureaucracy and complexity to the process. Further, although an Org-ID based approach for documenting force structure is planned in the U.S. Army, it does not yet exist. The original plan assumed that Org-IDs would be in place before an EID system was developed. This would allow one to exploit the Org-ID dissemination process. Clearly, this will not occur and it is an unnecessary requirement for implementing an EID scheme.

Second, it may not be appropriate that the only criteria for establishing an ES be the acquisition of an Org-ID. This allows anyone with an Org-ID to establish an ES, regardless of their need or position. Although this might be considered a good feature by some, it could complicate the task of configuration management. Typically, it is system developers that make decisions concerning data management policy. Therefore, requiring registration with an ESS adds a level of control to the authorization process. However, the Org-ID approach can still be implemented. If system managers decide that they want the ability to use any Org-ID as an EID seed, all that is required is to have the force development community obtain an EID seed for each new organization it creates. The 64-bit Org-EID would then simply be a 32 bit EID seed followed by a predefined, standard suffix. [14] The only restriction would be that an ES could not use the standard suffix.[15] This is illustrated in Figure 9.

Finally, one of the primary reasons for EIDs is to develop a scheme that is stable and easily scalable. Tightly coupling EIDs with Org-IDs does not contribute to this objective. Force structure is semi-static, stationary data, not static, reference data; it will change. This means that Org-IDs will come and go as organizations are established and disestablished. For example, the billet a person occupies[16] may be significantly modified requiring a new Org-ID. However, this is a documentation change that could have no effect on the physical world. The same person could still be executing the same duties with the same equipment. If EIDs were tightly coupled with Org-IDs, this would require changing the EID seed even though this is unnecessary from an information systems perspective. Why add this unnecessary complexity and aggravation?

One argument for using EID seeds for Org-IDs is that the EID seed can be used to identify the organization that created the data. This is not a strong argument for two reasons. First, a basic characteristic of EIDs is that they have no information encoded in them. The reason for using EID seeds is only to ensure uniqueness, not identify the creator. If identifying the creator of the data is required, then this should be included in the data schema (e.g., a field is included that, using an EID, identifies the record creator). Second, if required, an EID identification feature can be implemented via the registration process in the current approach. What would be required is the inclusion of the optional tracking service in the ESs. This is a uncomplicated task. In summary, it is recommended that EID seeds not be tightly coupled to Org-IDs as was originally recommended.

---

[14] An obvious choice is 32 zeros. For the Joint Common Database, the value 1 was chosen.

[15] This means that an EID server can not use the value selected as the predefined 32 bit suffix. This minutely reduces the number of available EIDs by one (i.e., to 4.3 billion – 1) for each EID server.

[16] A billet is also an organization with an Org-ID; see [2].

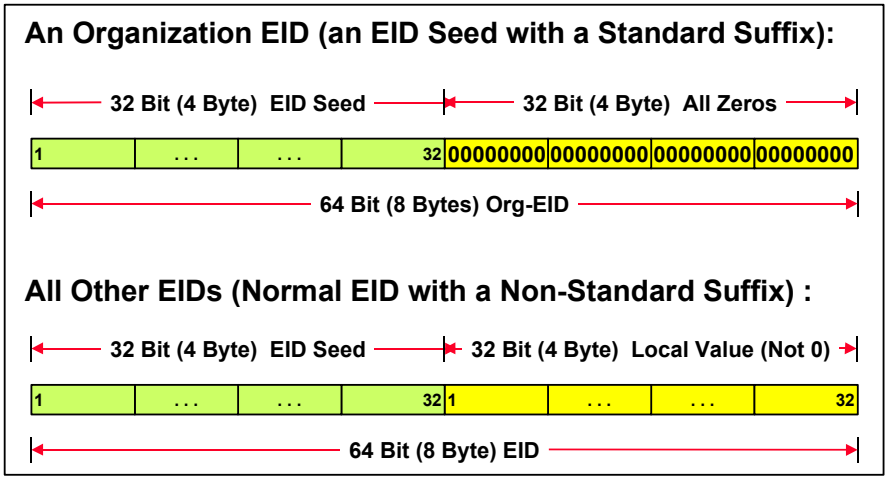

Figure 9: Org-EIDs Versus Other EIDs

4.2 Future Implementation Strategy

Much of the work to implement an EID seed server has already been accomplished. Its predecessor, the prototype Org-ID server, was established in April 1999 via a project with the U.S. Army ODISC4[17] Architecture Directorate. It has all the basic functionality of the ESS, but without a browser interface to facilitate the registration process. This is because the Org-ID server was originally designed to interface directly with other computer programs via a network, in this case, the prototype Army organization server.[18] Building a browser interface is a well-understood task and requires no additional study. However, firm policies have yet to be decided.

An ES can vary significantly in sophistication. At one extreme, it can be a clone of an ESS, and at the other extreme, it can be executed as a few lines of Java® code. There is currently no plan to define or build a standard set of ES APIs[19] or protocols; however, at least one will have to be defined when the Army Organization Server (AOS) is established.

The AOS is a requirement in the FMS ORD.[20] The current plan is for the AOS to be an output of the FMS. It will maintain the current Army force structure, down to the billet level, in a hierarchical structure that can be downloaded into C4I and other command information systems. It will contain only default (i.e., authorized) data and will not contain actual readiness information (this is maintained in other systems).

---

[17] Office of the Director of Information Systems for Command, Control, Communications and Computers

[18] See http://arch-odisc4.army.mil/Data_admn/html/org_id.htm.

[19] API: Application Program Interface: an interface specification to a software program; the well-defined set of operations by which the services supplied by other programmed components are accessed.

[20] FMS: Force Management System, a new, integrated force development system in the U.S. Army.
ORD: Operational Requirements Document – a user requirement specification document.

5. Summary

This paper describes a technique for providing a common naming convention for data, called enterprise identifiers (EID), that spans service, governmental and international boundaries. Several basic points were presented. First, in any automated information system (AIS), one of the most fundamental tasks is the unique identification of data items and components. Providing a common identification method is imperative for maintaining integrity within and across AISs (whether real C4ISR systems or simulations). Second, enterprise identifiers are one approach for providing unique values across disparate systems; they have a common structure and are unique across the enterprise. Third, a primary requirement for a successful EID allocation system is that it be flexible, and when required, completely distributed; it can not produce bottlenecks nor can it be the source of delay. Fourth, to accomplish these characteristics, a three tiered approach is proposed that is based upon the common idea of producing globally unique values by concatenating a locally managed unique value to an existing globally unique value; see Figure 10.

The advantages of using EIDs are clear, but there is one particularly difficult task that must be accomplished before this approach can be implemented: there must be an agreement within the enterprise to use them. Although this requires agreement on only a single field in a database, agreeing on anything is usually a challenge within the data standards community. One recommended strategy is to *not* make the use of EIDs mandatory. Instead, it could be a voluntary action. But every information system developer must contend with the data-naming problem. Perhaps no other single agreement can have a larger payoff than agreeing on this very basic feature of all data.

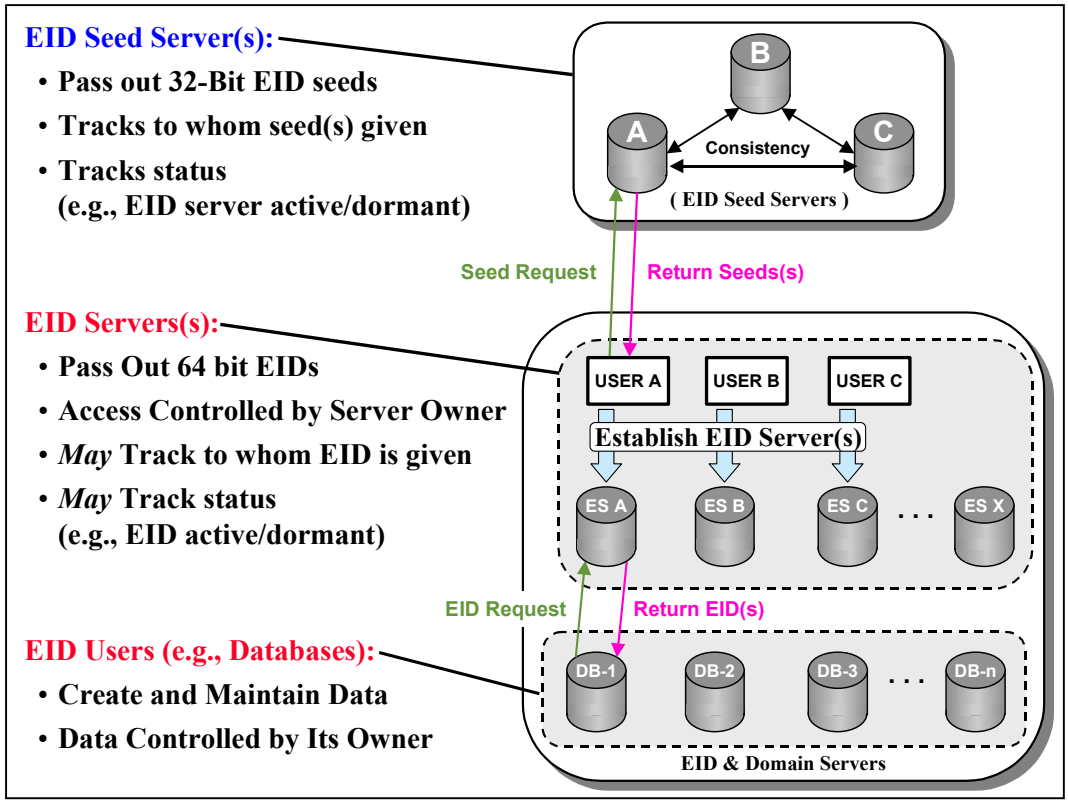

Figure 10: An EID Allocation Architecture

6. References

[1] M. Boller, "Common Understanding for Transformation Brigades," *Military Review*, Sep-Oct 2000. http://www-cgsc.army.mil/milrev/English/SepOct00/boller.htm

[2] S. Chamberlain: "Default Operational Representations of Military Organizations," Army Research Laboratory Technical Report: ARL-TR-2172; February 2000. http://www.arl.army.mil/~wildman/PAPERS/tr2172.html

[3] E.F. Codd: "Extending the Relational Model to Capture More Meaning,'" *ACM Transactions on Database Systems, Vol 4(4)*, Dec 1979.

[4] C.J. Date: "Relational Databases: Selected Writings," Addison-Wesley, Reading, MA, 1986.

[5] T. Johnston: "Primary Key Reengineering Projects: The Problem;" *DM Review*, February, 2000, http://www.dmreview.com/master.cfm?NavID=55&EdID=1866.

[6] T. Johnston: "Primary Key Reengineering Projects: The Solution;" *DM Review*, March, 2000, http://www.dmreview.com/master.cfm?NavID=55&EdID=2004

[7] T. Johnston: "De-Embedding Foreign Keys," *DM Direct* (online):
Part 1, June 2, 2000.
http://www.dmreview.com/editorial/dmreview/print_action.cfm?EdID=2308.
Part 2, June 9, 2000.
http://www.dmreview.com/editorial/dmreview/print_action.cfm?EdID=2322.
Part 3, June 16, 2000.
http://www.dmreview.com/editorial/dmreview/print_action.cfm?EdID=2331.
Part 4, June 23, 2000.
http://www.dmreview.com/editorial/dmreview/print_action.cfm?EdID=2341.

[8] M. Lonigro, Mike: "The Case for the Surrogate Key", *Intelligent Enterprise Database Programming and Design On-line,* May 1998
http//:www.dbpd.com/vault/9805xtra.htm.

[9] *Enterprise Identifiers For Logistics - An Approach in Support of Army Transformation Initiatives*,
http://arch-odisc4.army.mil/Data_admn/html/docs/ArmyLogisticsStudy_xFinal.pdf